Creating a Dream Team from T20 World Cup Data

Introduction

Four years ago, the ICC T20 World Cup took place, and England emerged victorious. As we reflect on the tournament's rich history, we're presented with a fascinating opportunity to revisit the performances of the world's top cricketers. Our goal is to create a dream team of the best 11 players, capable of achieving remarkable feats on the field. Specifically, our team should be able to score at least 180 runs on average and defend a total of 150 runs on average.

Data Analysis Process

To accomplish this task, we'll employ a rigorous data analysis process, involving the following steps:

1. Data Collection: We'll start by collecting data from the T20 World Cup matches held 4 years ago using web scraping techniques.
2. Data Cleaning and Transformation: Next, we'll clean and transform the collected data using Jupyter, ensuring that it's in a suitable format for analysis.
3. Data Transformation: We'll then transform the data using Power Query to prepare it for modeling and building.
4. Data Modeling and Building: Using DAX, we'll create a data model that captures the key performance indicators of each player, enabling us to identify the best 11 players for our dream team.
5. Dashboard Building: Finally, we'll build a comprehensive dashboard using Power BI to visualize our findings and showcase the performance of our dream team.

Conclusion

By leveraging the power of data analysis, we'll uncover the most exceptional players and build a team that would dominate the T20 format. This project will demonstrate the potential of data-driven decision-making in sports, providing valuable insights for coaches, players, and fans alike.

Data Collection:

T20 World Cup Player Data from ESPN Cricinfo

Objective

Our objective is to collect comprehensive data on players from the T20 World Cup held 4 years ago, using ESPN Cricinfo as our primary data source. We will utilize Bright Data's web crawling capabilities to extract relevant data from the website, which will be used to create a dream team of the best 11 players.

Web Page Inspection

Upon inspecting the ESPN Cricinfo web page dedicated to the T20 World Cup, we identified that each player's data is presented in separate table formats. The tables contain the following information:

* Batting Summary: A breakdown of the player's batting performance, including runs scored, average, strike rate, and more.
* Bowling Summary: A summary of the player's bowling performance, including wickets taken, economy rate, average, and more.
* Match Result: The outcome of each match, including the result, score, and opposition.
* Player Info/Stats: General information about the player, including their name, nationality, age, and career statistics.

Web Crawler Configuration

To extract the required data, we configured a web crawler using Bright Data's platform. The crawler is designed to navigate the ESPN Cricinfo website, identify the relevant tables, and extract the data into a JSON format.

Crawler Settings

* Start URL: The URL of the T20 World Cup page on ESPN Cricinfo.
* Selectors: CSS selectors used to identify the tables containing the player data.
* Extractors: Configured to extract the data from the identified tables and convert it into a JSON format.
* Rotation: Set to rotate user agents and IP addresses to avoid being blocked by the website.

Data Extraction

The web crawler successfully extracted the data from the ESPN Cricinfo website, resulting in a comprehensive JSON file containing the following data points for each player:

* Batting Summary:
  + Runs scored
  + Average
  + Strike rate
  + Hundreds
  + Fifties
* /\* -------------- STAGE 1 ------------ \*/
* //------- 1.a Interaction Code ------ //
* navigate('https://stats.espncricinfo.com/ci/engine/records/team/match\_results.html?id=14450;type=tournament');
* let links = parse().matchSummaryLinks;
* for(let i of links) {
* next\_stage({url: i})
* }
* //------- 1.b Parser Code ------------//
* let links = []
* const allRows = $('table.engineTable > tbody > tr.data1');
* allRows.each((index, element) => {
* const tds = $(element).find('td');
* const rowURL = "https://www.espncricinfo.com" +$(tds[6]).find('a').attr('href');
* links.push(rowURL);
* })
* return {
* 'matchSummaryLinks': links
* };
* /\* -------------- STAGE 2 ------------ \*/
* //------- 2.a Interaction Code ------ //
* navigate(input.url);
* collect(parse());
* //------- 2.b Parser Code ------------//
* var match = $('div').filter(function(){
* return $(this)
* .find('span > span > span').text() === String("Match Details")
* }).siblings()
* team1 = $(match.eq(0)).find('span > span > span').text().replace(" Innings", "")
* team2 = $(match.eq(1)).find('span > span > span').text().replace(" Innings", "")
* matchInfo = team1 +  ' Vs ' + team2
* var tables = $('div > table.ci-scorecard-table');
* var firstInningRows = $(tables.eq(0)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 8
* })
* var secondInningsRows = $(tables.eq(1)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 8
* });
* var battingSummary = []
* firstInningRows.each((index, element) => {
* var tds = $(element).find('td');
* battingSummary.push({
* "match": matchInfo,
* "teamInnings": team1,
* "battingPos": index+1,
* "batsmanName": $(tds.eq(0)).find('a > span > span').text().replace(' ', ''),
* "dismissal": $(tds.eq(1)).find('span > span').text(),
* "runs": $(tds.eq(2)).find('strong').text(),
* "balls": $(tds.eq(3)).text(),
* "4s": $(tds.eq(5)).text(),
* "6s": $(tds.eq(6)).text(),
* "SR": $(tds.eq(7)).text()
* });
* });
* secondInningsRows.each((index, element) => {
* var tds = $(element).find('td');
* battingSummary.push({
* "match": matchInfo,
* "teamInnings": team2,
* "battingPos": index+1,
* "batsmanName": $(tds.eq(0)).find('a > span > span').text().replace(' ', ''),
* "dismissal": $(tds.eq(1)).find('span > span').text(),
* "runs": $(tds.eq(2)).find('strong').text(),
* "balls": $(tds.eq(3)).text(),
* "4s": $(tds.eq(5)).text(),
* "6s": $(tds.eq(6)).text(),
* "SR": $(tds.eq(7)).text()
* });
* });
* return {"battingSummary": battingSummary}

Bowling Summary:

* + Wickets taken
  + Economy rate
  + Average
  + Best bowling figures
* /\* -------------- STAGE 1 ------------ \*/
* //------- 1.a Interaction Code ------ //
* navigate('https://stats.espncricinfo.com/ci/engine/records/team/match\_results.html?id=14450;type=tournament');
* let links = parse().playersLinks;
* for(let i of links) {
* next\_stage({url: i})
* }
* //------- 1.b Parser Code ------------//
* let links = []
* const allRows = $('table.engineTable > tbody > tr.data1');
* allRows.each((index, element) => {
* const tds = $(element).find('td');
* const rowURL = "https://www.espncricinfo.com" +$(tds[6]).find('a').attr('href');
* links.push(rowURL);
* })
* return {
* 'playersLinks': links
* };
* /\* -------------- STAGE 2 ------------ \*/
* //------- 2.a Interaction Code ------ //
* navigate(input.url);
* collect(parse());
* //---------- 2.b Parser Code ---------//
* var match = $('div').filter(function(){
* return $(this)
* .find('span > span > span').text() === String("Match Details")
* }).siblings()
* team1 = $(match.eq(0)).find('span > span > span').text().replace(" Innings", "")
* team2 = $(match.eq(1)).find('span > span > span').text().replace(" Innings", "")
* matchInfo = team1 + ' Vs ' + team2
* var tables = $('div > table.ds-table');
* var firstInningRows = $(tables.eq(1)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 11
* })
* var secondInningsRows = $(tables.eq(3)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 11
* });
* var bowlingSummary = []
* firstInningRows.each((index, element) => {
* var tds = $(element).find('td');
* bowlingSummary.push({
* "match": matchInfo,
* "bowlingTeam": team2,
* "bowlerName": $(tds.eq(0)).find('a > span').text().replace(' ', ''),
* "overs": $(tds.eq(1)).text(),
* "maiden": $(tds.eq(2)).text(),
* "runs": $(tds.eq(3)).text(),
* "wickets": $(tds.eq(4)).text(),
* "economy": $(tds.eq(5)).text(),
* "0s": $(tds.eq(6)).text(),
* "4s": $(tds.eq(7)).text(),
* "6s": $(tds.eq(8)).text(),
* "wides": $(tds.eq(9)).text(),
* "noBalls": $(tds.eq(10)).text()
* });
* });
* secondInningsRows.each((index, element) => {
* var tds = $(element).find('td');
* bowlingSummary.push({
* "match": matchInfo,
* "bowlingTeam": team1,
* "bowlerName": $(tds.eq(0)).find('a > span').text().replace(' ', ''),
* "overs": $(tds.eq(1)).text(),
* "maiden": $(tds.eq(2)).text(),
* "runs": $(tds.eq(3)).text(),
* "wickets": $(tds.eq(4)).text(),
* "economy": $(tds.eq(5)).text(),
* "0s": $(tds.eq(6)).text(),
* "4s": $(tds.eq(7)).text(),
* "6s": $(tds.eq(8)).text(),
* "wides": $(tds.eq(9)).text(),
* "noBalls": $(tds.eq(10)).text()
* });
* });
* return {"bowlingSummary": bowlingSummary}

Match Result:

* + Result (win/loss/tie)
  + Score
  + Opposition
* /\* -------------- STAGE 1 ------------ \*/
* //------- 1.a Interaction Code ------ //
* navigate('https://stats.espncricinfo.com/ci/engine/records/team/match\_results.html?id=14450;type=tournament');
* collect(parse());
* //------- 1.b Parser Code ------------//
* //Step1: create an array to store all the records
* let matchSummary = []
* //Step2: Selecting all rows we need from target table
* const allRows = $('table.engineTable > tbody > tr.data1');
* //Step3: Looping through each rows and get the data from the cells(td)
* allRows.each((index, element) => {
* const tds = $(element).find('td');   //find the td
* matchSummary.push({
* 'team1':  $(tds[0]).text(),
* 'team2':  $(tds[1]).text(),
* 'winner':  $(tds[2]).text(),
* 'margin':  $(tds[3]).text(),
* 'ground': $(tds[4]).text(),
* 'matchDate': $(tds[5]).text(),
* 'scorecard':   $(tds[6]).text()
* })
* })
* // step4: Finally returning the data
* return {
* "matchSummary": matchSummary};

Player Info/Stats:

* + Name
  + Nationality
  + Age
  + Career statistics (runs scored, wickets taken, etc.)
* /\* -------------- STAGE 1 ------------ \*/
* //------- 1.a Interaction Code ------ //
* navigate('https://stats.espncricinfo.com/ci/engine/records/team/match\_results.html?id=14450;type=tournament');
* let links = parse().matchSummaryLinks;
* for(let i of links) {
* next\_stage({url: i})
* }
* //------- 1.b Parser Code ------------//
* let links = []
* const allRows = $('table.engineTable > tbody > tr.data1');
* allRows.each((index, element) => {
* const tds = $(element).find('td');
* const rowURL = "https://www.espncricinfo.com" +$(tds[6]).find('a').attr('href');
* links.push(rowURL);
* })
* return {
* 'matchSummaryLinks': links
* };
* /\* ------------ STAGE 2 -------------- \*/
* //------- 2.a Interaction Code ------ //
* navigate(input.url);
* let playersData = parse().playersData;
* for(let obj of playersData) {
* name = obj['name']
* team = obj['team']
* url = obj['link']
* next\_stage({name: name, team: team, url: url})
* }
* //---------- 2.b Parser Code ---------//
* //to store all the players in a list
* var playersLinks = []
* var match = $('div').filter(function(){
* return $(this)
* .find('span > span > span').text() === String("Match Details")
* }).siblings()
* team1 = $(match.eq(0)).find('span > span > span').text().replace(" Innings", "")
* team2 = $(match.eq(1)).find('span > span > span').text().replace(" Innings", "")
* //for batting players
* var tables = $('div > table.ci-scorecard-table');
* var firstInningRows = $(tables.eq(0)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 8
* })
* var secondInningsRows = $(tables.eq(1)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 8
* });
* firstInningRows.each((index, element) => {
* var tds = $(element).find('td');
* playersLinks.push({
* "name": $(tds.eq(0)).find('a > span > span').text().replace(' ', ''),
* "team": team1,
* "link": "https://www.espncricinfo.com" + $(tds.eq(0)).find('a').attr('href')
* });
* });
* secondInningsRows.each((index, element) => {
* var tds = $(element).find('td');
* playersLinks.push({
* "name": $(tds.eq(0)).find('a > span > span').text().replace(' ', ''),
* "team": team2,
* "link": "https://www.espncricinfo.com" + $(tds.eq(0)).find('a').attr('href')
* });
* });
* //for bowling players
* var tables = $('div > table.ds-table');
* var firstInningRows = $(tables.eq(1)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 11
* })
* var secondInningsRows = $(tables.eq(3)).find('tbody > tr').filter(function(index, element){
* return $(this).find("td").length >= 11
* });
* firstInningRows.each((index, element) => {
* var tds = $(element).find('td');
* playersLinks.push({
* "name": $(tds.eq(0)).find('a > span').text().replace(' ', ''),
* "team": team2.replace(" Innings", ""),
* "link": "https://www.espncricinfo.com" + $(tds.eq(0)).find('a').attr('href')
* });
* });
* secondInningsRows.each((index, element) => {
* var tds = $(element).find('td');
* playersLinks.push({
* "name": $(tds.eq(0)).find('a > span').text().replace(' ', ''),
* "team": team1.replace(" Innings", ""),
* "link": "https://www.espncricinfo.com" + $(tds.eq(0)).find('a').attr('href')
* });
* });
* return {"playersData": playersLinks}

* /\* ------------- STAGE 3 ------------ \*/
* //------- 3.a Interaction Code ------ //
* navigate(input.url);
* final\_data = parse()
* collect(
* {
* "name": input.name,
* "team": input.team,
* "battingStyle": final\_data.battingStyle,
* "bowlingStyle": final\_data.bowlingStyle,
* "playingRole":  final\_data.playingRole,
* "description": final\_data.content,
* });
* //---------- 3.b Parser Code ---------//
* const battingStyle = $('div.ds-grid > div').filter(function(index){
* return $(this).find('p').first().text() === String('Batting Style')
* })
* const bowlingStyle = $('div.ds-grid > div').filter(function(index){
* return $(this).find('p').first().text() === String('Bowling Style')
* })
* const playingRole = $('div.ds-grid > div').filter(function(index){
* return $(this).find('p').first().text() === String('Playing Role')
* })
* return {
* "battingStyle": battingStyle.find('span').text(),
* "bowlingStyle": bowlingStyle.find('span').text(),
* "playingRole": playingRole.find('span').text(),
* "content": $('div.ci-player-bio-content').find('p').first().text()
* }

**Next Steps**

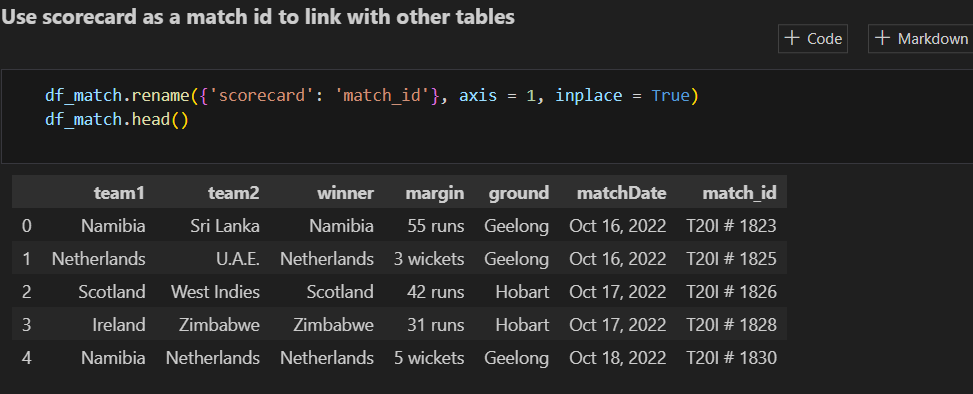
The extracted data will be used to create a Power Query, which will enable us to transform and model the data to identify the best 11 players for our dream team. The Power Query will be built using Microsoft Power BI, allowing us to visualize and analyze the data to make informed decisions.

**DATA CLEANING and Transformation**

This Jupyter notebook is used to clean and transform the T20 World Cup cricket data extracted from ESPN Cricinfo using Bright Data's web crawler. The data is stored in JSON files and will be processed into CSV files for further analysis.

**Section 1: Process Match Results**

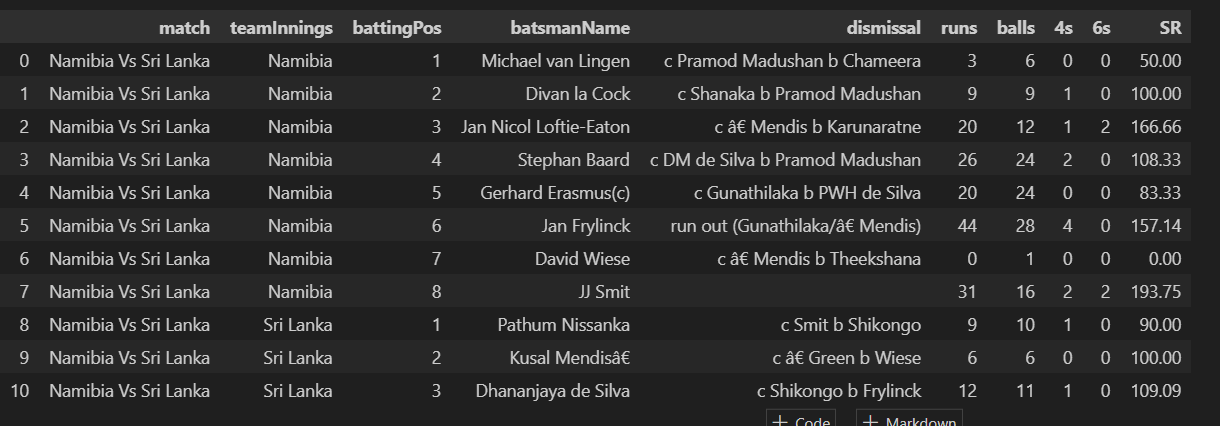
* Load the **t20\_wc\_match\_results.json** file into a pandas DataFrame **df\_match**.
* Rename the **scorecard** column to **match\_id** to use as a unique identifier for each match.
* Create a dictionary **match\_ids\_dict** to map team names to a unique match ID.
* Save the processed data to **dim\_match\_summary.csv**.





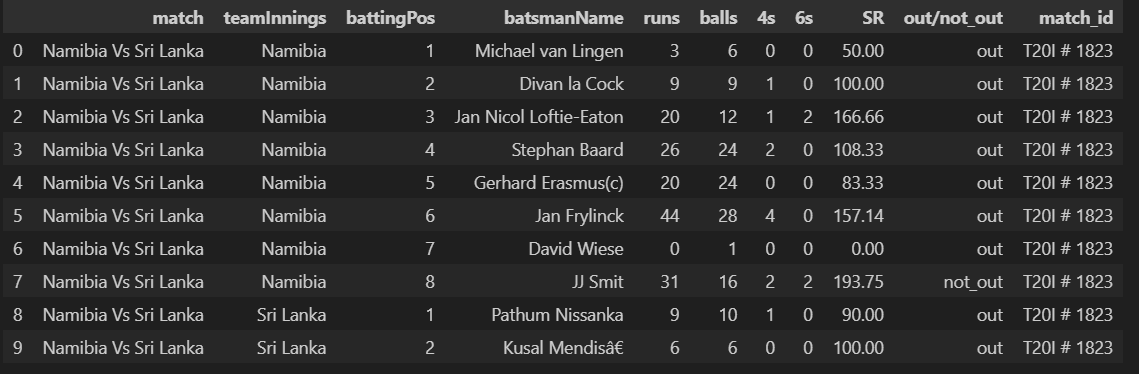
**Section 2: Process Batting Summary**

* Load the **t20\_wc\_batting\_summary.json** file into a pandas DataFrame **df\_batting**.
* Extract the batting summary records from the JSON data.
* Add a new column **out/not\_out** based on the **dismissal** column.
* Map the **match** column to the **match\_id** using the **match\_ids\_dict**.



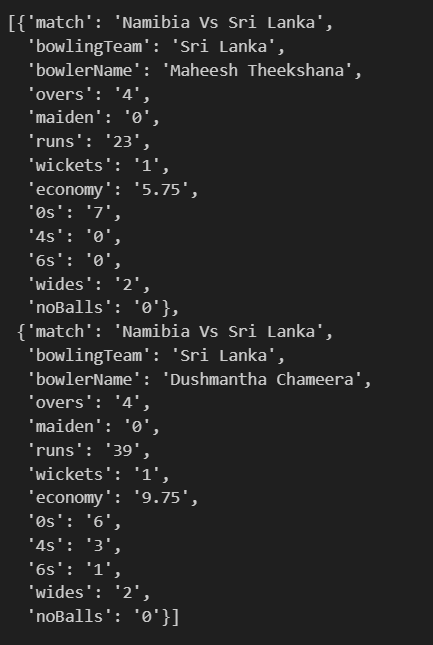


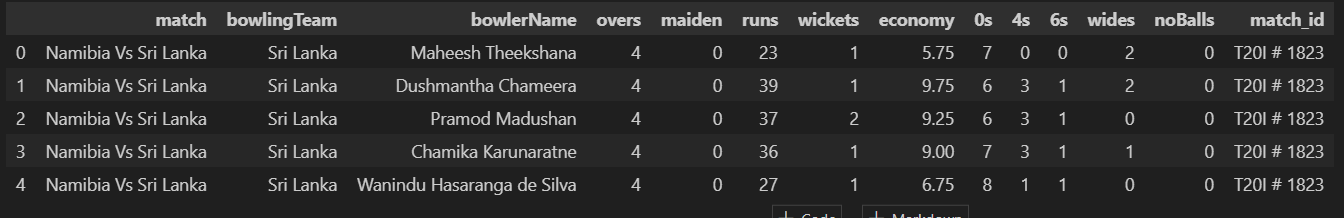
* Drop the **dismissal** column.
* Clean up weird characters in the **batsmanName** column.
* Save the processed data to **fact\_bating\_summary.csv**.



**Section 3: Process Bowling Summary**

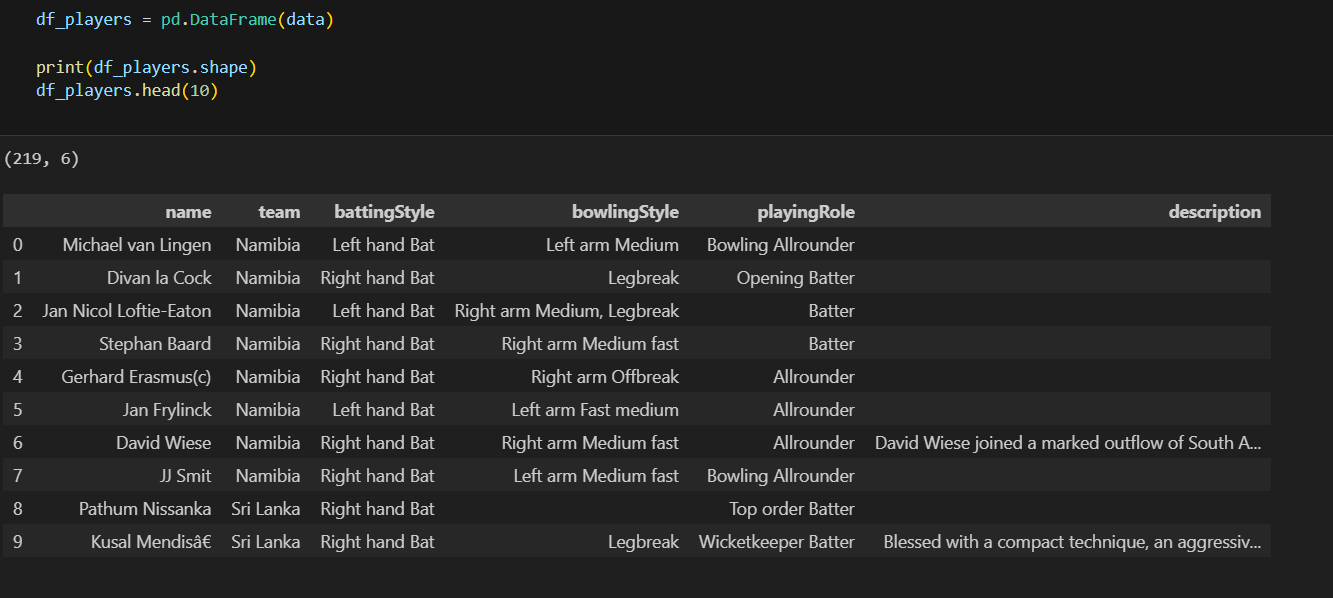
* Load the **t20\_wc\_bowling\_summary.json** file into a pandas DataFrame **df\_bowling**.
* Extract the bowling summary records from the JSON data.
* Map the **match** column to the **match\_id** using the **match\_ids\_dict**.
* Save the processed data to **fact\_bowling\_summary.csv**.

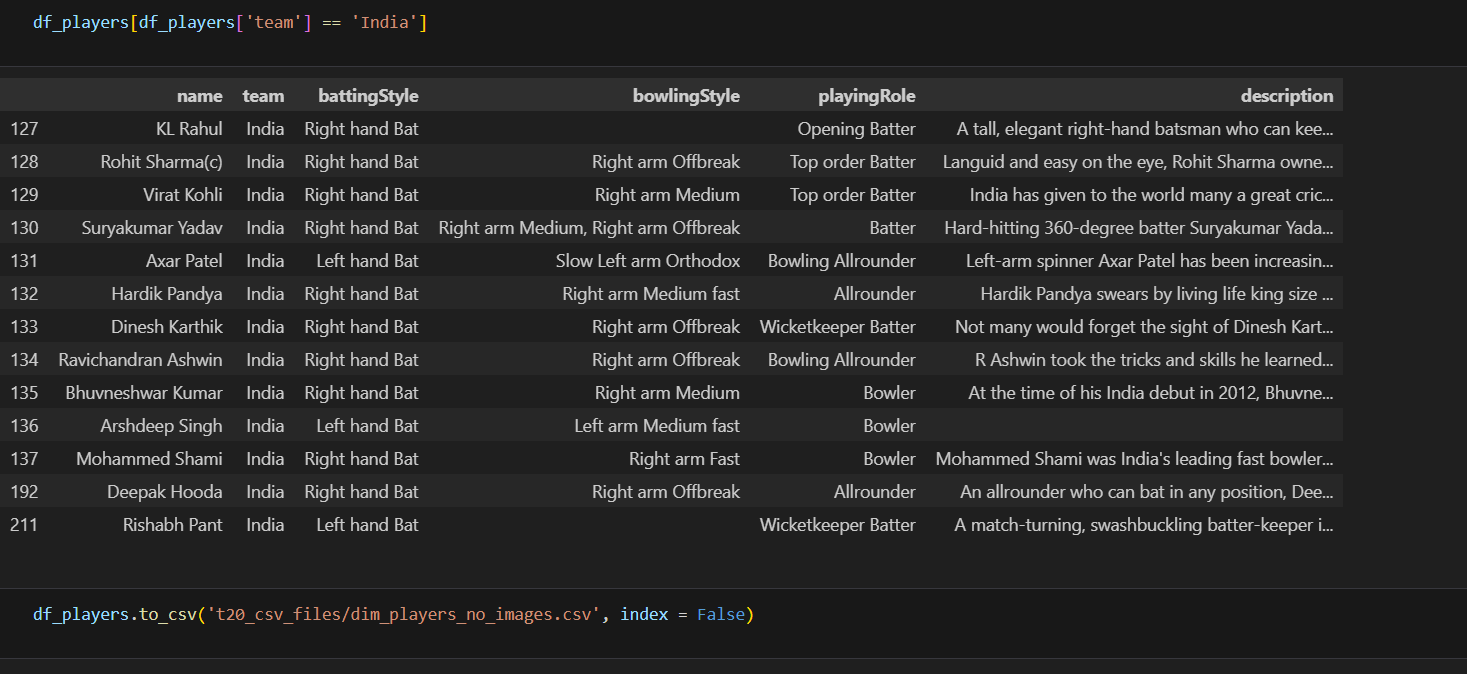




**Section 4: Process Players Information**

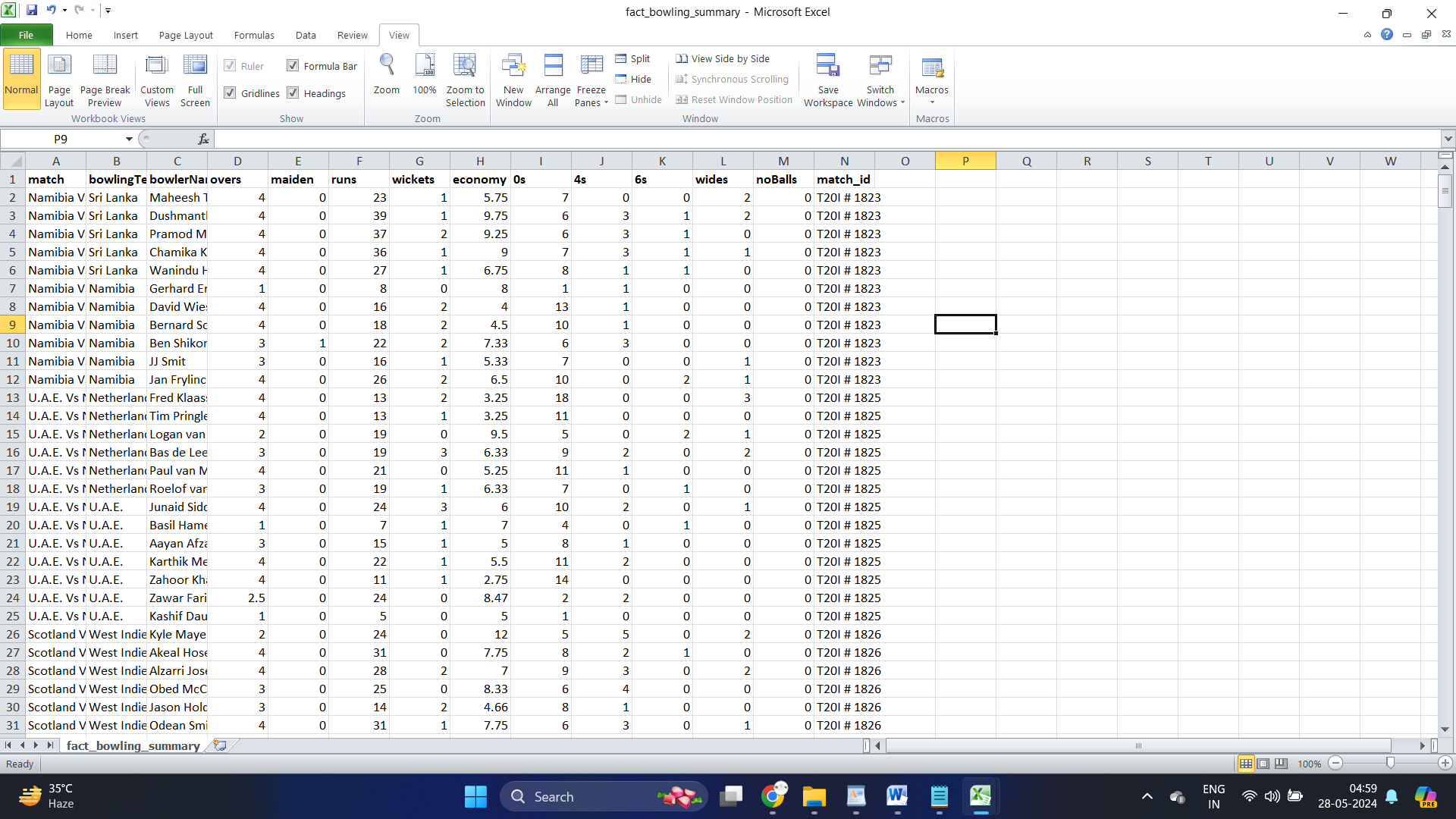
* Load the **t20\_wc\_player\_info.json** file into a pandas DataFrame **df\_players**.
* Clean up weird characters in the **name** column.
* Save the processed data to **dim\_players\_no\_images.csv**

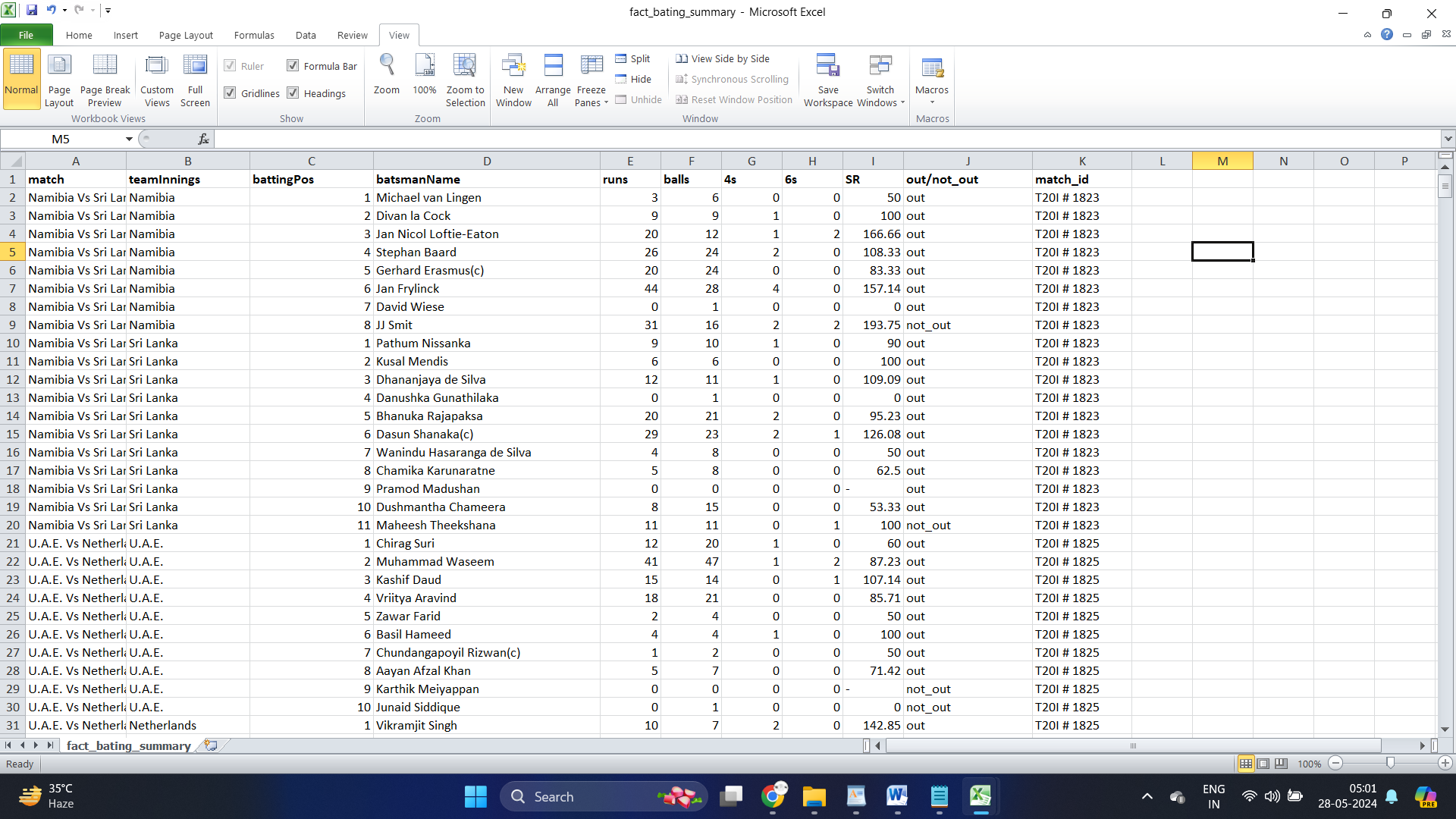


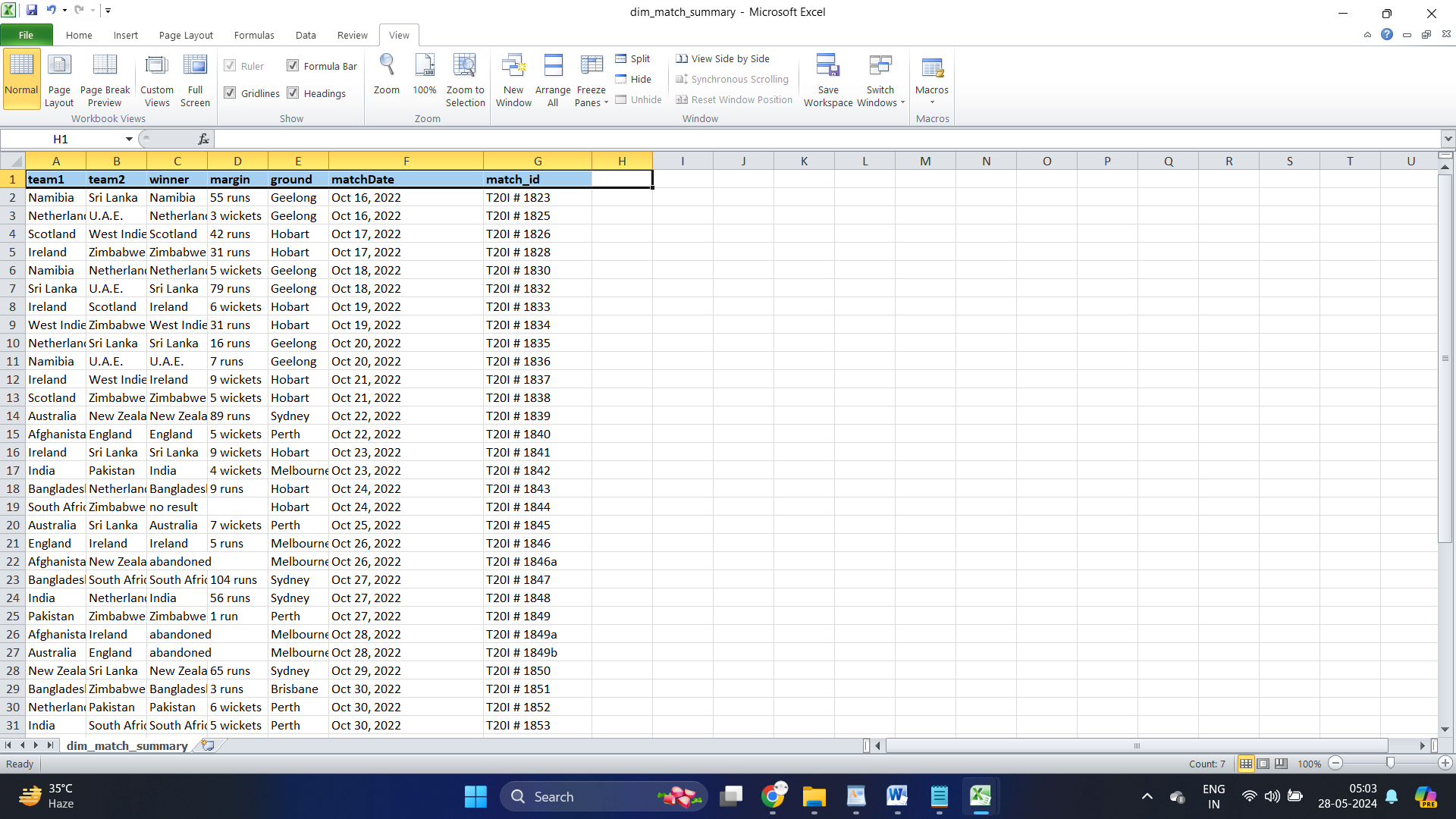
****

**data transformation in (power query)**

After scraping the data from ESPNcricinfo using Bright Data's data collector tool, the next step is to clean and transform the data using Power Query in Excel. The image shows how to transform the "match" column using the "mapping" function in Power Query. By mapping the "match" column, the user is able to create a new column called "match ID" that links the two tables together.

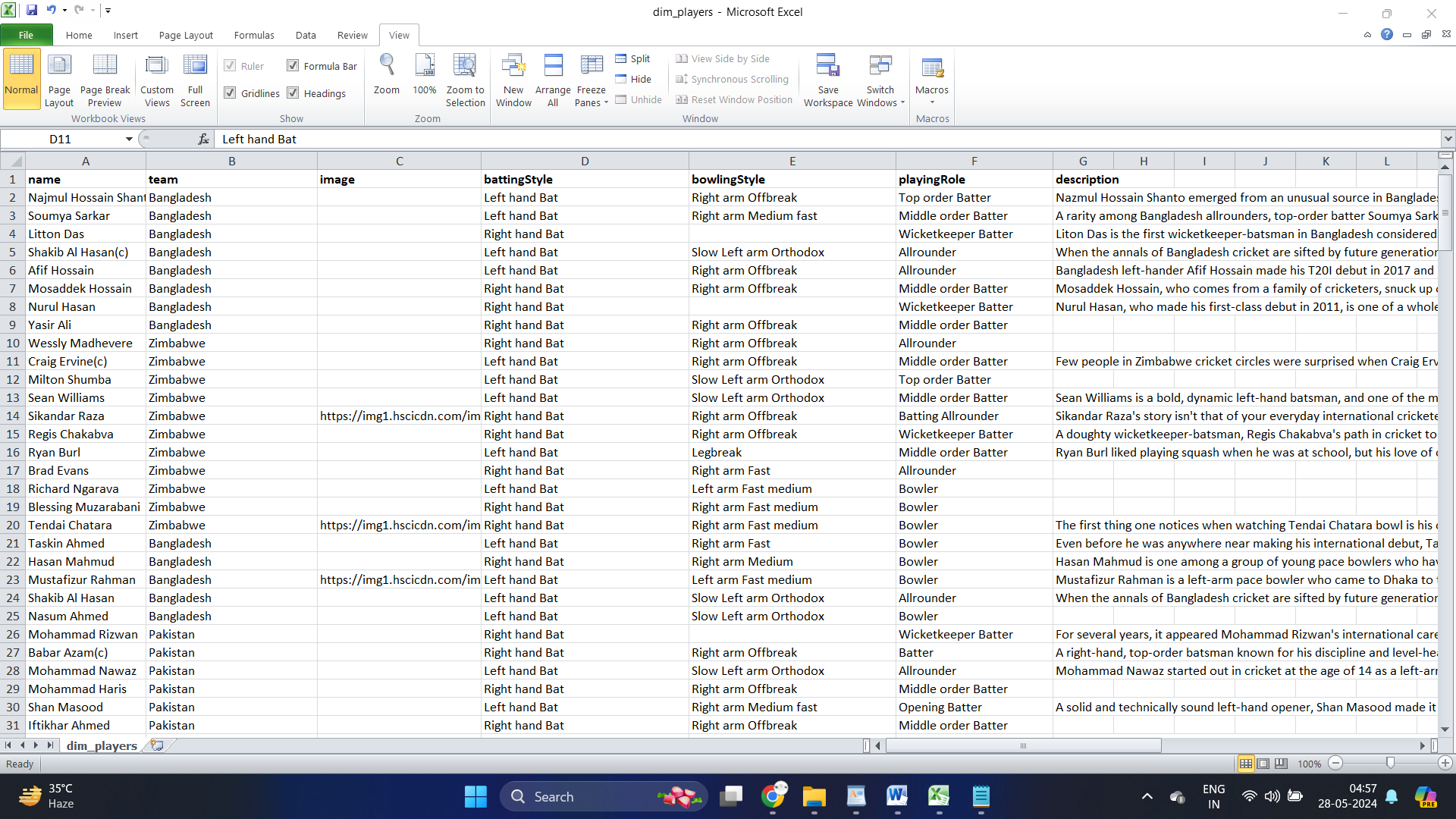






then moves on to show the entire Power Query notebook, which includes the transformation of the match results, batting summary, bowling summary, and player information. At each point, the user exports the file as a CSV.

The player information is saved in two separate files: "dim players.csv" and "no images.csv". The "dim players.csv" file includes an extra column for the player's image, which has been manually added to the file. The video shows a preview of the "dim players.csv" file, which includes columns for the player name, team, batting style, bowling style, player role, and description. The "image" column contains a link to the player's image.



Overall, the image demonstrates the process of cleaning and transforming data using Power Query in Excel. By mapping the "match" column, the user is able to link two tables together and create a new column called "match ID". The transformed data is then exported as a CSV file, which can be used for further analysis. The also shows how to manually add images to the player information file, which can be useful for visualizing the data.

**data modeling and building parameters using DAX**

DAX (Data Analysis Expressions) is a formula language used in Microsoft Power BI, Power Pivot for Excel, and SQL Server Analysis Services (SSAS) Tabular models. It is used to define calculations and measures in a data model, allowing users to perform complex calculations and data analysis.

DAX is a powerful and flexible language that provides a wide range of functions and operators for working with data. It includes functions for aggregating data, calculating ratios and percentages, working with dates and times, and filtering data. With DAX, users can create calculated columns, measures, and tables that can be used to analyze and visualize data in a variety of ways.



The measures include:

* Total Runs: The sum of runs scored by a batsman.
* Total Innings Batted: The number of innings a batsman has batted in.
* Total Innings Dismissed: The number of innings a batsman has been dismissed in.
* Batting Average: The average number of runs scored per innings by a batsman.
* Total balls Faced: The total number of balls faced by a batsman.
* Strike Rate: The rate at which a batsman scores runs per 100 balls faced.
* Batting Position: The average batting position of a player.
* Boundary %: The percentage of runs scored by a batsman that are boundaries (fours or sixes).
* Avg. balls Faced: The average number of balls faced by a batsman per innings.
* Wickets: The total number of wickets taken by a bowler.
* balls Bowled: The total number of balls bowled by a bowler.
* Runs Conceded: The total number of runs conceded by a bowler.
* Bowling Economy: The average number of runs conceded per over by a bowler.
* Bowling Strike Rate: The number of balls bowled per wicket taken by a bowler.
* Bowling Average: The average number of runs conceded per wicket by a bowler.
* Total Innings Bowled: The total number of innings a bowler has bowled in.
* Dot Ball %: The percentage of balls bowled by a bowler that did not result in a run being scored.
* Player Selection: A measure that returns "1" if a player is selected and "0" otherwise.
* Display Text: A measure that displays a message if no player is selected.
* Color Callout Value: A measure that changes the color of a value based on whether a player is selected.

These measures can be used to analyze individual player performance, compare players, and identify trends in cricket matches. They can also be used to create visualizations and reports to communicate insights and support decision-making.



Calculated columns are a powerful feature in Power BI that allow users to create new columns in a table based on existing columns. These new columns can be used to perform calculations, transformations, and data manipulation.

The calculated columns listed above are used to perform specific calculations and transformations on the data in the **fact\_batting\_summary** and **dim\_player** tables.

Here is a brief description of each calculated column:

1. **boundary runs**: This calculated column calculates the total number of runs scored by a batsman through boundaries (fours and sixes) by multiplying the number of fours by 4 and the number of sixes by 6.
2. **Boundary runs bowling**: This calculated column calculates the total number of runs conceded by a bowler through boundaries (fours and sixes) by multiplying the number of fours by 4 and the number of sixes by 6.
3. **Custom Batting Order**: This calculated column assigns a custom batting order to a list of specific players based on their names. The **SWITCH** function is used to evaluate the player's name and assign a corresponding batting order.

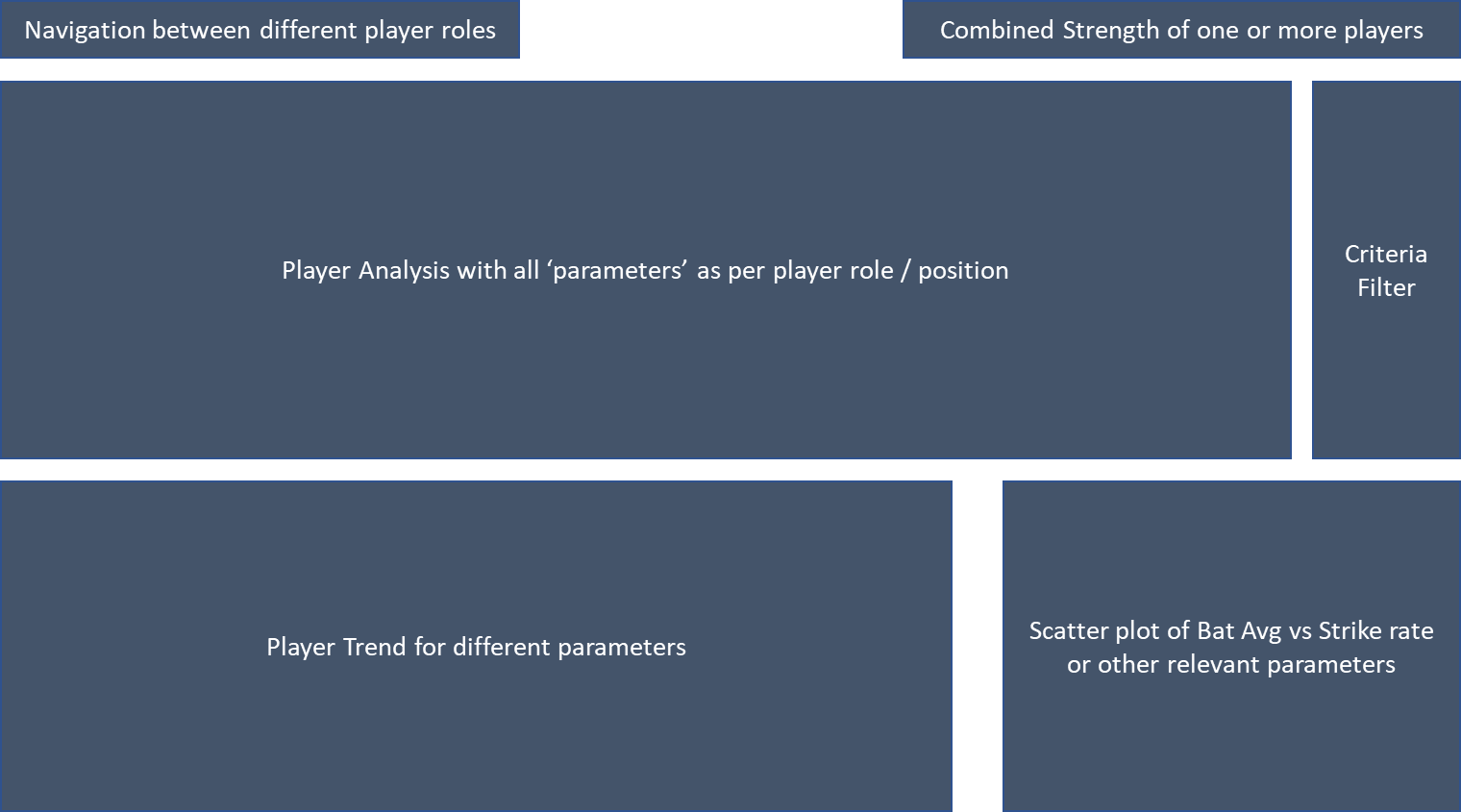
These calculated columns can be used to gain insights into the performance of batsmen and bowlers, and to create custom visualizations and reports.

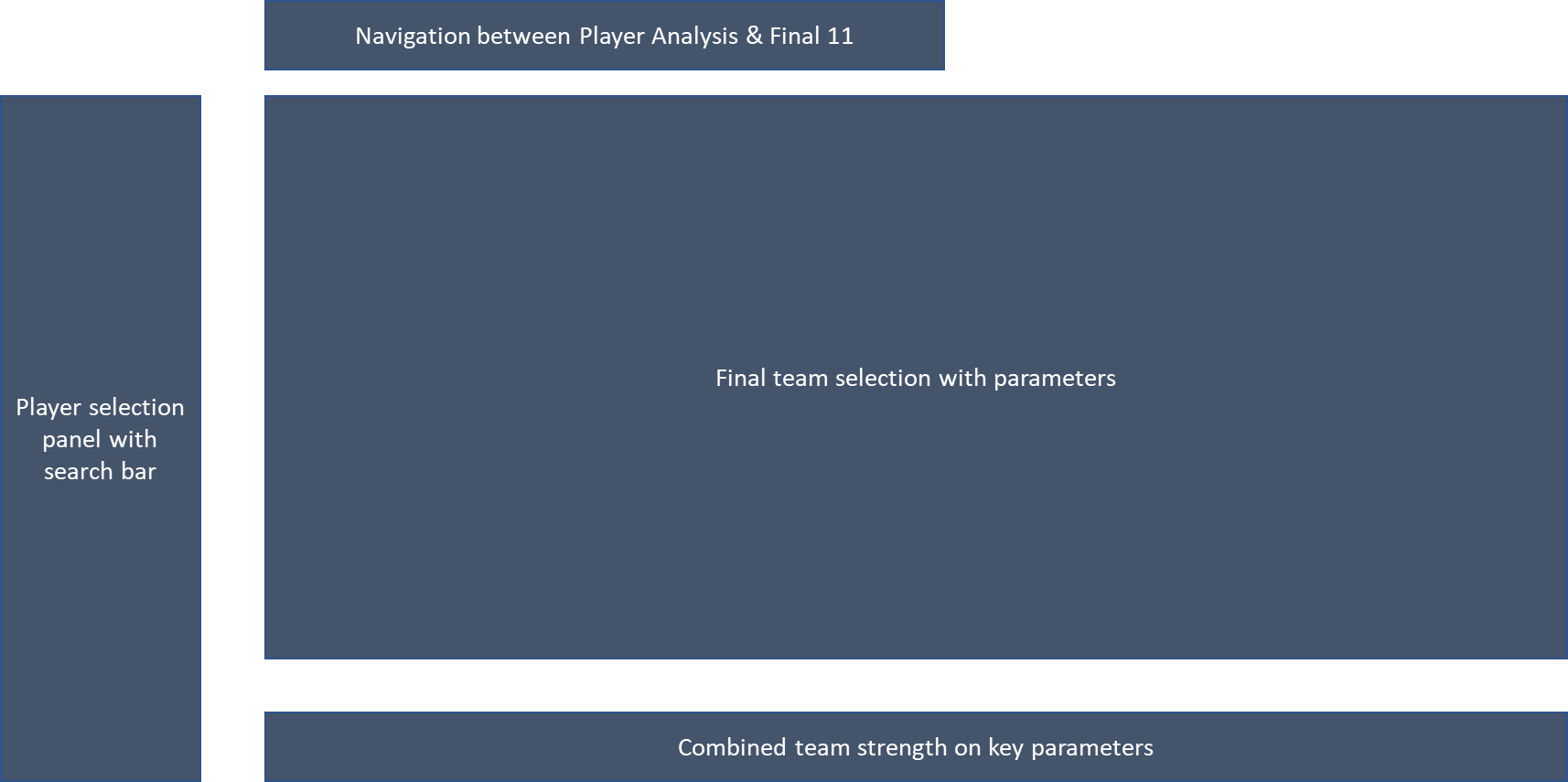
**building dashboard using power BI**

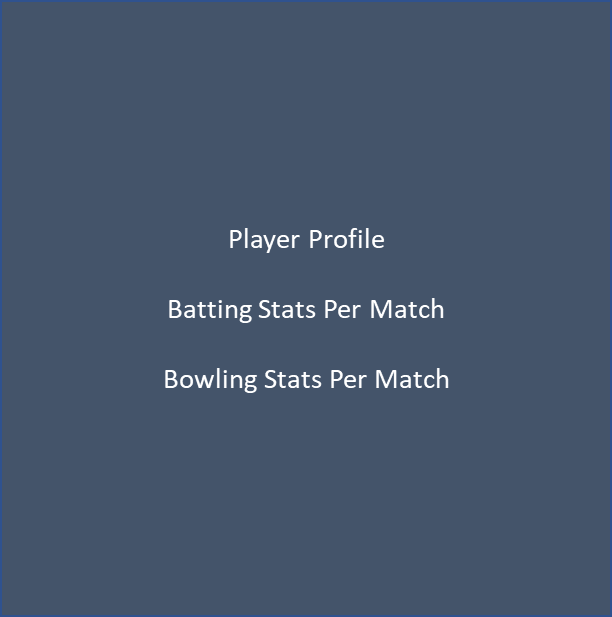
Power BI is a business analytics tool developed by Microsoft that provides interactive visualizations and business intelligence capabilities with an interface simple enough for end users to create their own reports and dashboards. It allows users to connect to a wide range of data sources, including Excel spreadsheets, databases, and cloud-based services, and to transform and clean the data for analysis.

Power BI provides a suite of tools for data modeling, visualization, and reporting, including Power Query for data transformation, Power Pivot for data modeling, and Power View for data visualization. It also includes a range of pre-built visualizations, such as charts, tables, and maps, as well as the ability to create custom visualizations using the Power BI API.

One of the key features of Power BI is its ability to create interactive dashboards and reports that can be shared with others. Users can create custom visualizations and reports, and then publish them to the Power BI service, where they can be accessed by others in the organization. Power BI also includes collaboration features, such as the ability to comment on reports and dashboards, and to share them with others.

****

****

****

1. **Color Selection**

**Primary and/or secondary colors**

1. Choose from the corporate identity e.g., LOGO
2. Choose from the industry. Example: For the metal industry choose grey and yellow for beauty products choose purple/ red/ pink, for fashion brands dark blue or black, etc.
3. Choose colors based on the theme of the dashboard or report

**Theme colors**

Determine the colors for the various elements on the dashboard such as page background, container background, borders, primary texts, secondary texts, highlighting texts, etc.

**Font Selection**

Choose the fonts for the identity/industry/subject.

**Size**

Determining the size for the KPI values, title text, value texts, and paragraph texts (in case of storytelling reports) in realtion to their importance and intended visibiltiy.

**Aspect Ratio**

Never go out of aspect ratios, it will make your static elements/ visuals ugly.

1. Try to keep canvas size 16:9 ratio (popular ratios 1024×576, 1280×720, 1600×900, 1920×1080, 2560×1440).

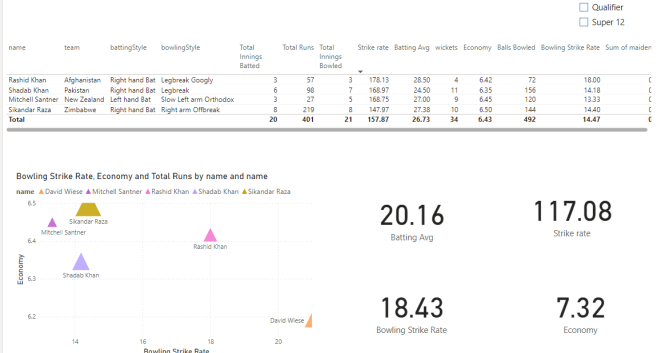
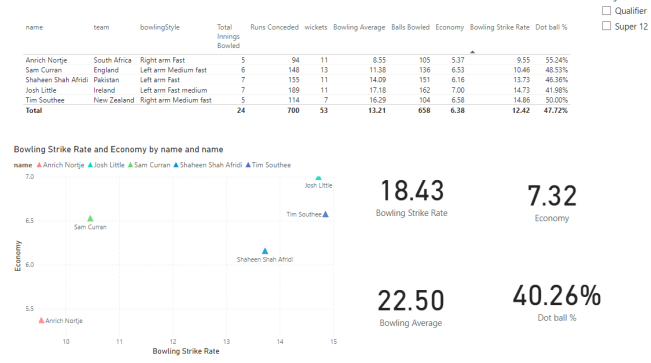
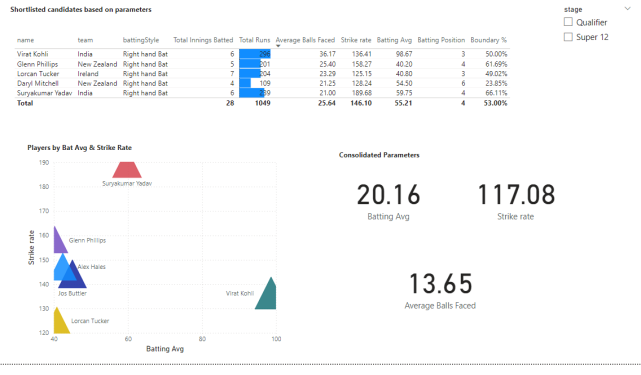
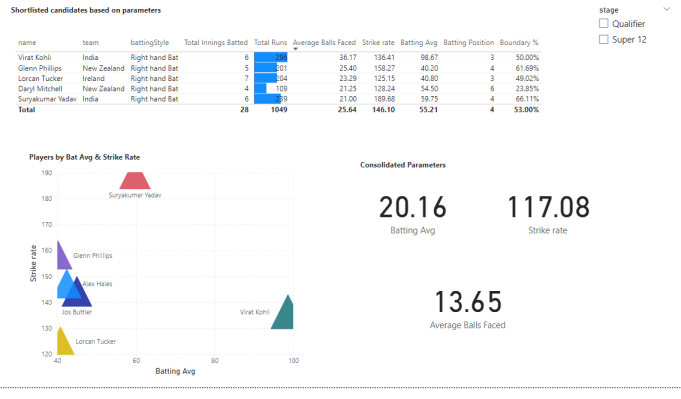
In case of custom size keep width either 1024, 1280, 1600, 1920, or 2560 and keep Height/ Length as required to accommodate all visuals

1. Strictly keep Images & LOGOs in their aspect ratio 3. Try to keep other elements’ size in multiple of 8 or if needed in a few cases in multiples of 4.

**4. Positioning**

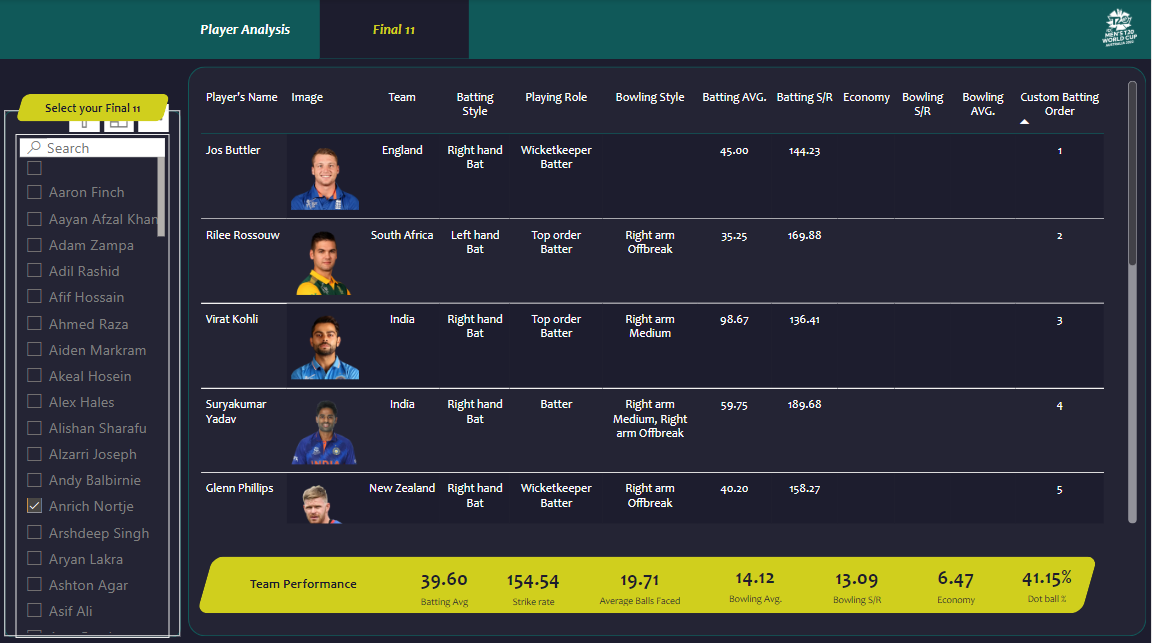
1. Try to place the elements at the positions (Horizontal / Vertical) in a multiple of 8 and f need than a multiple of 4

2. Maintain proper distances (in most cases equal) from left and right to each other - not too congested or not too open

****

**Final Dashboard**

****

****

****

****

**Summary**

**roject Title:** Dream Team Selection Dashboard using Power BI

**Objective:** To design and develop a dashboard that helps users select the best Dream players team based on their stats, using data from the ESPN Cric website of T20 World Cup.

**Project Overview:**

Our project involved a comprehensive approach to data analysis and visualization, leveraging various tools and technologies to create a user-friendly dashboard. The project consisted of eight stages:

1. **Introduction**: Defined the project scope and objectives.
2. **Pre-requisites**: Identified the necessary tools and technologies required for the project.
3. **Requirements and Scoping**: Determined the data requirements and scope of the project.
4. **Data Collection**: Utilized Bright Data for web scraping to collect data from the ESPN Cric website of T20 World Cup.
5. **Data Cleaning and Transformation**: Cleaned and transformed the data using Jupyter to prepare it for analysis.
6. **Data Transformation in Power Query**: Further transformed the data in Power Query to create a structured dataset.
7. **Data Modeling and Building Parameters using DAX**: Created data models and built parameters using DAX to enable data analysis and visualization.
8. **Building Dashboard using Power BI**: Designed and developed a dashboard in Power BI to visualize the data and enable users to select the best Dream players team based on their stats.

**Key Features:**

* The dashboard provides an interactive and user-friendly interface to select players based on their performance stats.
* Users can filter and sort players by various criteria, such as batting average, bowling average, and fielding performance.
* The dashboard visualizes complex data in an easy-to-understand format, enabling users to make informed decisions.
* The project demonstrates the power of data analysis and visualization in simplifying complex data representation.

**Benefits:**

* The dashboard helps users to select the best Dream players team, increasing their chances of winning.
* The project showcases the capabilities of Power BI in data visualization and analysis.
* The dashboard can be used as a template for similar projects, demonstrating the versatility of the approach.

Overall, our project successfully created a dashboard that simplifies complex data representation and enables users to make informed decisions when selecting a Dream players team.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |